Software–Projekt 2007/08

VAK 03-05-G-901.01

JAKARTA

Software Development

Testplan

Version 1.0

Artur Malek	amalek@tzi.de	2152026
Yasin Ünsal	yasinu@tzi.de	1874544
Levent Özenen	levent85@tzi.de	2131928
Sascha Schmidt	rone1983@web.de	2174364

Abgabe: 21. April 2008

Inhaltsverzeichnis

0 Version und Änderungsgeschichte

(bearbeitet von: Artur Malek)

0.1 Version

Öffentliche Version 1.0
Interne Version 0.8

0.2 Änderungsgeschichte

Version	Änderungen
0.1	Punkt 1 hinzugefügt
0.2	Punkt 3 hinzugefügt
0.3	Punkt 7 hinzugefügt
0.4	Punkt 6 hinzugefügt
0.5	Punkt 4 hinzugefügt
0.6	Punkt 5 hinzugefügt
0.7	Punkt 2 hinzugefügt
0.8.1	Punkt 8.1 hinzugefügt
1.0	Die erste Veröffentlichung

1 Einführung

(erstellt von: Artur Malek)

Bei diesem Dokument handelt es sich um den Testplan für das Projektmanagementsystem, welches von der Gruppe Jakarta Software Development im Rahmen des Softwareprojekts 2007/2008 an der Universität Bremen erstellt wird.

1.1 Zweck

(erstellt von: Artur Malek)

Der Zweck des Testplans ist es, ein strukturiertes und systematisches Testen unseres Systems zu ermöglichen. Im Testplan werden die Komponenten, die zu testen sind, festgelegt und es werden für diese Komponenten geeignete Testfälle entworfen. Ausserdem muss festgelegt werden, in was für einem Umfang getestet wird und wann ein Test als erfolgreich durchgeführt gilt und wann der Test abgebrochen werden muss. Der Testplan definiert zudem auch, welche Hard- und Software und andere Bertriebsmittel benötigt werden, um die Tests durchführen zu können. Dieses Dokument ist für alle Mitarbeiter des Projekts und den Kunden gedacht, insbesondere richtet sich der Testplan aber an die Tester des Systems, da sie auf der Grundlage dieses Dokuments das System testen sollen.

1.2 Umfang

(erstellt von: Artur Malek)

Der Inhalt und Aufbau des Testplanes entspricht der vereinfachten Form des *IEEE Standard 829-1998*, welcher in der von uns besuchten Vorlesung des Software-Projekts vorgestellt wurde. In diesem Dokument werden hier nur die Integrations- und Leistungstests beschreiben. Sie sollen der Schwerpunkt der Testfälle sein. Eine genaue Beschreibung der Komponententests ist in unserem Testplan nicht enthalten. Die Komponententests werden als JUnit-Tests erstellt und ihre Spezifikation geht daraus hervor.

1.3 Beziehungen zu anderen Dokumenten

(erstellt von: Artur Malek)

Zwischen dem Testplan und der Anforderungsspezifikation und der Architekturbeschreibung bestehen Beziehungen. Diese werden nun im folgenden beschreiben.

1.3.1 Anforderungsspezifikation

Wie schon kurz erwähnt, besteht zwischen dem Testplan und der Anforderungsspezifikation eine Beziehung. Der Testplan dient dazu, Testfälle zu entwickeln, die auf den Anwendungsfällen der Anforderungsspezifikation basieren. Es muss festgestellt werden, wo kritische Bereiche des Systems liegen und wie diese durch Tests überprüft werden können, um die Funktionalität

du gewährleisten, die mit dem Kunden vereinabrt wurde. Ziel der Tests ist es natürlich, dem Kunden ein fehlerfreies und stabil laufendes System auszuliefern.

1.3.2 Architekturbeschreibung

Der Testplan basiert auch auf der Architekturbeschreibung unseres Systems. In der Architekturbeschreibung wurden die Komponenten beschreiben, die jetzt getestet werden sollen. Die Komponenten wurden dort detailiert beschrieben und die Bezeichnungen der Komponenten werden im Testplan übernommen. Für jede Komponente beschreiben wir, wie und in was für einem Umfang sie getestet werden soll.

1.4 Aufbau der Testbezeichner

(erstellt von: Artur Malek)

Für den Aufbau der Testbezeichner benutzen wir den im Beispieldokument genannten Aufbau der Testbezeichner.
(siehe http://www.informatik.uni-bremen.de/st/Lehre/swp/testplan.pdf)

Dieser setzt sich folgendermassen zusammen:

- Die ersten beiden Buchstaben stehen für die Art des Tests (s. Tab. 1)
- Die Nummer steht für die Testfallnummer der gewählten Testart
- Der Buchstabe am Ende steht für die Variante des Tests oder für einen Untertestfall. Dies muss aber nicht angegeben werden, wenn es keine Varianten oder Untertestfälle gibt (z.b. A für erste Variante, D für vierte Variante)

Ein bespielhafter Testbezeichner wäre folgender:

LT-1-B: Leistungstest, Nummer 1, Variante 2

LT	Leistungstest
IT	Integrationstest
FT	Funktionstest

Tabelle 1 : Abkürzungen der Testbereiche

1.5 Definitionen und Akronyme

(erstellt von: Artur Malek)

- JVM: Java Virtual Machine, Java Laufzeitumgebung
- Server: Ein Server ist ein Programm, das mit einem anderen Programm, dem Client, kommuniziert, um ihm Zugang zu speziellen Dienstleistungen zu verschaffen. (de.wikipedia.org/Server)
- IEEE: Institute of Electrical and Electronics Engineers, Normung von Hardware, Software und Techniken
- s.: siehe
- Tab.: Tabelle
- z.B.: zum Beispiel

1.6 Referenzen

(erstellt von: Artur Malek)

- Aufbau des Testplans:
  http://www.informatik.uni-bremen.de/st/Lehre/swp/abgabe4.html

- IEEE Standard:
  IEEE Standard 830-1998: IEEE Standard for Software Test Documentation

- Inhaltliche Struktur:
  http://www.informatik.uni-bremen.de/st/Lehre/swp/abgabe4.html
  Vorlesung Software-Projekt Universität Bremen 07/08

2 Systemüberblick

(bearbeitet von: Sascha Schmidt)

Um den Funktionsumfang zu gewährleisten, der mit dem Kunden in der Anforderungsspezifikation festgelegt wurde, basiert der Testplan auf der Anforderungsspezifikation. Ausserdem basiert der Testplan auf der Architekturbeschreibung. Dort wurden bereits alle zu testenden Komponenten festgelegt und beschreiben. Um sicherzustellen, dass die Software fehlerfrei und stabil ist, wird genau festgelegt wie und in welchem Umfang getestet wird.
Das Projekt-Management-System setzt sich aus 4 Komponenten zusammen:

1. ProjectManagement
Die ProjectManagement Komponente beinhaltet die gesammte Funktionalität zum Verwalten von Projekten, Arbeitspaketen, Personen und Berichten. Diese Komponente ist im Aufbau zwar recht simpel, stellt gleichzeitig aber die wichtigste Komponente dar, da das gesamte System darauf aufbaut. Das bedeutet, dass diese Komponente besonders ausführlich getestet wird und die Test eine besonders hohe Wertigkeit geniessen, weil Fehler in diesem Bereich das ganze system gefährden.

2. Netzwerkkomponente
Diese Komponente ist in sofern von der Project Komponente abhängig, da sie auf die Projektdaten zugreift und per TCP/IP an den Server sendet. Die Daten müssen also korrekt vorliegen. Ausserdem muss das Senden und Empfangen der Daten ausführlich getestet werden, da dies den Mehrbenutzer Betrieb gewährleistet, welcher Teil der Anforderungen ist und somit eine auch eine hohe Priorität geniesst.

3. Datenbank Komponente
Die Datenbank Komponente läuft auf dem Server und beinhaltet alle Datenbankabfragen. Diese komponente ist zwar so wie die Netzwerkkomponente in Bezug auf die Funktionalität in sich geschlossen, setzt aber trotzdem eine Fehlerfreihe Netzwerkkomponente voraus, da diese verwendet wird um Daten zu empfangen(Queries) und zurück an den Client zu schicken.

4. XML komponente
Lokale Daten werden in XML-Dateien gespeichert. Die Funktionalität dafür befindet sich in dieser Komponente.

3 Merkmale

(bearbeitet von: Artur Malek)

In diesem Abschnitt werden die zu testenden und die nicht zu testenden Merkmale aufgeführt. Es wird ausreichend erklärt, warum einige Merkmale nicht getestet werden müssen.

3.1 Zu testende Merkmale

(bearbeitet von: Artur Malek)

In diesem Abschnitt werden die Merkmale aufgeführt, die getestet werden sollen. Die Klassen, die wir implementieren, müssen funktionieren, damit wir gewährleisten, dass das System korrekt funktioniert. Auch muss die Kommunikation zwischen einzelnen Klassen oder Komponenten korrekt funktionieren. Es muss getestet werden, ob die Module so implementiert sind und auch so funktionieren, wie sie in der Anforderungsspezifikation spezifiziert wurden. Fehler, die beim Testen auftreten, müssen korrgiert werden, bis das System sich korrekt verhält. Ausserdem muss das System sich geeignet in Fehlerfällen wie einer falschen Eingabe verhalten. Um dies alles zu erreichen, sollten geeignete Testfälle entwickelt werden.

1. ProjectManagement

    1.1 Report

        1.1.2 Einen Bericht zu einem Arbeitspaket erstellen
        1.1.3 Einen Bericht zu einem Arbeitspaket löschen
        1.1.4 Einen Bericht zu einem Arbeitspaket bearbeiten

    1.2 Person

        1.2.1 Eine Person erstellen
        1.2.2 Eine Person löschen
        1.2.3 Eine Person als Administrator festlegen
        1.2.4 Die Daten einer Person bearbeiten

1.3 Allocation

1.3.1 Eine Zuweisung einer Person zu einem Arbeitspaket erstellen

1.3.2 Eine Zuweisung einer Person zu einem Arbeitspaket löschen

1.3.3 Eine Zuweisung einer anderen Person zu einem Arbeitspaket geben

1.3.4 Eine Zuweisung für mehrere Personen zu einem Arbeitspaket erstellen

1.3.5 Eine Zuweisung für eine Person löschen (wenn mehrere Personen dran arbeiten)

1.4 WorkingPackage

1.4.1 Ein Arbeitspaket erstellen

1.4.2 Ein Arbeitspaket löschen

1.4.3 Den Status eines Arbeitspakets festlegen (begonnen, nicht begonnen, beendet)

1.4.4 Ein Unterarbeitspaket erstellen

1.4.5 Ein Unterarbeitspaket löschen

1.4.6 Status eine Unterarbeitspakets festlegen

1.4.7 Daten des Arbeitspakets aändern (Name, Beschreibung usw.)

1.5 Project

1.5.1 Ein Projekt erstellen

1.5.2 Ein Projekt löschen

1.5.3 Person dem Projekt hinzufügen

1.5.4 Person aus dem Projekt löschen

1.5.5 Arbeitspaket dem Projekt hinzufügen

1.5.6 Arbeitspaket aus dem Projekt löschen

1.5.7 Bericht dem Projekt hinzufügen

1.5.8 Bericht aus dem Projekt löschen

1.5.9 Zuweisung dem Projekt hinzufügen

1.5.10 Zuweisung aus dem Projekt löschen

1.6 SQLQueries

1.6.1 Anfrage an den Server übertragen

1.6.2 Angeforderte Daten zu einer Funktion übertragen

1.6.3 Anmelden am Server

1.6.4 Abmelden am Server

2.0 XML Interface

2.0.1 Projekt lokal speichern

2.0.2 Projekt lokal laden

2.0.3 XML-Datei auslesen

3.0 Datenbank

Datenbankinhalt sichern (Backup)

Datenbankinhalt wiederherstellen

3.2 Nicht zu testende Merkmale

Hier werden die nicht zu testenden Merkmale aufgeführt. Warum sie nicht getestet werden müssen, wir detailiert erklärt. Verwendete Java-Bibliotheken, die wir nicht selbst erstellt haben, müssen nicht von uns getestet werden, da sie schon von den Entwicklern getestet wurden. Das gleiche gilt für Server-Software, die wird im Zusammenhang mit der Datenbank verwenden werden.

1. ProjectManagement

1.1 Report

Bericht zu einem nicht vorhandenen Arbeitspaket erstellen:
Dies muss nicht getestet werden, da man über das Kontextmenü nach einem Rechtsklick auf ein Arbeitspaket einen Bericht erstellen kann. Ist das Arbeitspaket nicht vorhanden, kann man dies auch nicht aufrufen.

Bericht zu einem nicht vorhandenen Arbeitspaket löschen:
Die Möglichkeit, einen Bericht zu löschen, besteht nur, wenn das Kontextmenü geöffnet ist. So ist dies auch wie beim Erstellen des Berichts nicht möglich, dies aufzurufen, wenn dieses Arbeitspaket nicht existiert.

Bericht eines nicht vorhandenen Arbeitspakets bearbeiten:
Genauso wie in den beiden anderen Punkten, ist dies nur möglich, wenn das Kontextmenü offen ist.

1.2 Person

Nicht vorhandene Person löschen:
Dieser Testfall muss nicht berücksichtigt werden, weil man im Menü *Personen verwalten* nur Personen angezeigt bekommt, die in diesem Projekt mitarbeiten und folglich schon erstellt wurden. Damit ist dieser Testfall sinnlos.

Nicht vorhandene Person bearbeiten: Dies ist aus den gleichen Gründen wie beim Löschen einer nicht vorhandenen Person nicht möglich.

Nicht vorhandene Person als Administrator festlegen:
Dies ist auch nicht möglich, wegen den oben genannten Gründen. Um eine Person als Administrator festzulegen, muss man eine Checkbox als Checked setzen, die dann eine boolsche Variable auf true setzt. Ist eine Person nicht vorhanden, ist dies nicht möglich.

1.3 Allocation

Zuweisung eines Arbeitspakets zu einer nicht vorhandenen Person:
Dies ist nicht möglich, da man eine Person erstellt haben muss, um eine Zuweisung zu einem Arbeitspaket durchzuführen. Es werden dem Benutzer auch nur Personen zur Zuweisung angezeigt, die zum Projekt gehören und somit erstellt wurden.

Zuweisung von einer Person zu einem nicht vorhandenen Arbeitspaket:
Dies muss auch nicht getestet werden, da ein nicht erstelltes Arbeitspaket nicht angezeigt wird, um einen Bericht zu schreiben.

1.4 Workingpackage

Nicht vorhandenes Arbeitspaket löschen:

Nicht vorhandenes Arbeitspaket bearbeiten:
Beide Merkmale müssen nicht berücksichtigt werden. Nicht vorhandene Arbeitspakete werden nicht angezeigt und können somit nicht gelöscht oder bearbeitet werden.

1.5 Project

Nicht vorhandenes Projekt löschen:

Nicht vorhandenes Projekt bearbeiten:
Dies muss nicht berücksichtigt werden, da es nicht möglich ist, eine nicht existierendes Projekt zu löschen oder zu bearbeiten.

1.6 SQL-Queries

Anmelden ohne Passwort:

Anmelden ohne Benutzernamen:
Anmelden ohne Passwort bzw. ohne Benutzernamen wird nicht gehen, da es beides benötigt, um sich am Server anzumelden. Daher muss dies nicht getestet werden.

Datenanfrage senden ohne mit dem Server verbunden zu sein:
Um Daten von der Datenbank anzufordern oder sie zu verändern, ist es nötig, mit dem Server verbunden zu sein. Daher beschäftigen wir uns nicht mit diesem Merkmal.

2.0 XML Interface

lokales Speichern eines leeren Projekts:

Laden einer leeren XML-Datei:
Beide Fälle sind sinnlos, da wir in beiden Fällen eine leere XML-Datei hätten. Somit gäbe es nichts, was in eine XML-Datei zu speichern bzw. aus einer XML-Datei auszulesen wäre.

4 Abnahme- und Testendekriterien

(bearbeitet von: Sascha Schmidt)

4.1 Testabdeckung

Um ein stabiles System ausliefern zu können, muss die Testabdeckung so hoch wie möglich sein, so dass alle Fehler, die die Funktionalität beeinträchtigen, gefunden werden können. Dafür wird die Testabdeckung in unterschiedliche Bereiche unterteilt, wobei jeder Bereich eine andere Art zu Testen darstellt. Hier sei darauf hingewiesen, dass die Testabdeckungen entsprechend der Prioritäten der Komponenten variieren, da der Einfluss der Komponenten auf das gesamte System von Komponente zu Komponente unterschiedlich ist. Die unteren Schichten des Programms haben eine höhere Testabdeckung, da diese das gesamte System beeinflussen. Im Gegensatz dazu ist die Testabdeckung in Bereichen, die keinen groen Einfluss auf die Funktionalität haben, wesentlich geringer.
Die Testergebnisse werden in Fehlerklassen ,die einer bestimmten Wertigkeit zugeornet wird, kategorisiert.

Fehlerklasse 1
Beschreibung: unwesentlich: Fehler, der keine sichtbaren Schäden verursachen,aber trotzdem korrigiert werden müssen
Wertigkeit: 1

Fehlerklasse 2
Beschreibung: gering: Fehler, der das System in seiner Leistung einschränken oder dessen Verfügbarkeit reduzieren könnte
Wertigkeit: 10

Fehlerklasse 3
Beschreibung: kritisch: Fehler, der schwere Schäden am System verursachen können
Wertigkeit: 20

Fehlerklasse 4
Beschreibung: katastrophal: Fehler, der die Unterbrechung des Systems verursachen kann bzw. das System unbrauchbar macht
Wertigkeit: 100

4.2 Verhältnis Fehler / Lines-Of-Code

Das Verhältnis von Fehler zu Lines-Of-Code kann von Klasse zu Klasse stark variieren. Deshalb wird das Verhältnis auch nicht für jede Klasse einzeln festgelegt. Dies würde einen zu groen Aufwand bedeuten. Stattdessen werden die Fehler pro 1000 Zeilen Code betrachtet und unter Berücksichtigung der Fehlerklassen die Testergebnisse als Kriterien für die Testdauer herangezogen. Dabei dürfen pro 1000 Zeilen Code nur Fehler auftreten, die einen festgelegten Schwellwert nicht überschreiten. Ausserdem darf der Test nicht beendet werden, solange noch Fehler der Fehlerklassen 3 und 4 auftreten. Budget Da es sich um ein studentisches Projekt handelt, welches einer Frist unterliegt, ist unser Budget einer begrenzten Zeit gleichzusetzen. Die für die Testphase eingeplante Zeit ist so festgelegt, dass wir ausreichend viele Tests durchführen können, um ein stabiles Produkt ausliefern zu können. Jedoch ist die Zeitplanung knapp, weshalb wir die Priorität bei der Durchführung der Tests auf die Erkennung und Beseitung kritischer Fehler setzen.

5 Vorgehensweise

(bearbeitet von: Artur Malek)

In diesem Kapitel beschreiben wir unsere Vorgehenweise beim Testen. Im folgenden werden nun Black-Box-Tests, White-Box-Tests und die von uns gewählte Integrationsstrategie erläutert.

5.1 Black-Box-Tests

Die Black-Box-Tests können schon sehr früh in der Implementierung erstellt werden, vor der eigentlichen Implementierung auf Basis der Spezifikation. Sie werden ohne Kenntnis der inneren Funktionsweise des Systems entwickelt werden. Nachdem man die Tests erstellt hat und der Code vorliegt, kann man einen Black-Box-Test ausführen. Mit den Black-Box-Tests kann festgestellt werden, ob der bis dato erstellte Code schon fehlerfrei läuft. Sollte er dies nicht tun, wird der Code korrigiert, und zu einem anderen Zeitpunkt wieder durch den gleichen Black-Box-Test getestet. Damit ist es klar, das ein Black-Box-Test während der Implementierungsphase mehrmals auf einen Code ausgeführt wird. Ziel der Black-Box-Test ist es, dass das System mit der Spezifikation übereinstimmt.

5.2 White-Box-Tests

Die White-Box-Tests werden erst zu einem späteren Zeitpunkt entwickelt. Sie werden auf der Basis des implementierten Codes erstellt und können somit erst nach Abschluss der Implementierungsphase entwickelt werden. Sie dienen vorallem dazu bestimmte Module zu testen. Jeder Test, der einen Fehler aufdeckt, sorgt dafür, dass der Code angepasst oder geändert werden muss. Dementsprechend muss dann auch der White-Box-Test an den geänderten Code angepasst werden. Dadruch ensteht ein grösserer Aufwand im Gegensatz zu den Black-Box-Tests. Daher werden White-Box-Tests nicht so häufig eingesetzt wie Black-Box-Tests.

5.3 Integrationsstrategien

Wir haben für die Testphase die Integrationsstrategie Bottom-Up gewählt. Durch diese Strategie wird es uns möglich sein, einfach

zu verstehende Tests zu entwickeln, da wir auf das aufwendige Erstellen von benötigten Teststümpfen verzichten wollen. Dadruch erhoffen wir uns, deutlich schneller und effektiver zu testen.

6 Aufhebung und Wiederaufnahme

(bearbeitet von: Sascha Schmidt)

In diesem Abschnitt definieren wir Kriterien, die zur Unterbrechung des Tests führen, um wieder zur Implementationsphase zurück zu kehren. Auch werden die Kriterien festgelegt, die erfüllt werden müssen, um den Test wieder aufzunehmen. Um diesen Vorgang zu erleichtern, werden für die jeweiligen Testphasen und Schwere der Fehler Schwellwerte festgelegt. Auf diese Weise ist es auch möglich, die Qualität des Programms in Bezug auf die Stabilität zu bewerten. Da Fehler in den unteren Schichten des Programms eine gröere Auswirkung auf die Funktionsweise des Programms haben, als jene in oberen Schichten, bekommen die Tests in den unteren Schichten einen niedriegeren Schwellwert zugewiesen. Für die Durchführung der Tests wird JUnit unter NetBeans verwendet. Als Testplattformen kommen sowohl Windows und Linux, als auch MacOs zum Einsastz.

Komponente	Schwellwert	Beschreibung
ProjectManagement	100	niedriger SW, da gesamte Funktionalität darauf aufbaut
XMLInterface	300	hoher SW, da Funktionalität nicht davon abhängt
DBInterface	300	hoher SW, da Funktionalität nicht davon abhängt
GUI	200	mittlerer SW, da Funktionalität ohne GUI nicht nutzbar

Tabelle 1: Tabelle mit den Schwellwerten der einzelnen Komponenten

Im Falle eines Fehlers und Überschreitung des jeweiligen Schwellwertes wird der Test solange ausgesetzt, bis die notwendigen Änderungen am Code durchgeführt wurden. Auch bei kritischen Fehlern wird die Testphase solange ausgesetzt, bis die Fehler behoben wurden, selbst wenn dies z.B. im Falle einer Abweichung von der Spezifikation eine komplette Neuimplementation der Komponente erfordert. Bei der Wiederaufnahme der Tests ist unbedingt zu beachten, dass alle Tests für Komponenten die abhängig von veränderten Komponenten sind, wiederholt werden müssen! Erst

wenn sichergestellt wurde, dass alle Tests von abhängigen Komponenten eine Fehlerquote haben, die niedrieger ist als der entsprechende Schwellwert, darf der Test fortgesetzt werden.

7 Hardware- und Softwareanforderungen

(bearbeitet von: Artur Malek)

7.1 Hardware

An Hardware werden mehrere Rechner mit ausreichender Rechenkapazität und Rechenleistung benötigt. Rechner, die dem heutigen Standard entsprechen, sollten für unsere Zwecke reichen. Um die Tests durchzuführen benötigen wir einen Datenbankserver, um zu testen, ob unser System mit dem externen Datenbankserver läuft. Die Kommunikation der Komponenten erfolgt über eine TCP/IP-Verbindung. Um diese herzustellen, müssen unsere Rechner eine Netzwerkkarte besitzen. Um einen Zugriff zu ermöglichen, werden ausserdem noch Netzwerkkabel (Patch-Kabel) benötigt.

7.2 Software

Um zu testen, benötigen wir mehrere Rechner, auf denen ein Betriebssystem und die entsprechende JVM installiert ist. Um zu garantieren, das unser System portierbar ist, werden die Tests auf verschiedenen Betriebssystemen durchgeführt. Eine sehr hohe Portierbarkeit wird gewährleistet, wenn wir jeden Test auf den unterschiedlichen Betriebsystemen durchführen. Geplant sind Tests mit Windows, Linux und MacOS. Die Tests auf den windowsbasierten Rechnern werden von den Gruppenmitgliedern zu Hause durchgeführt. Die linuxbasierten und applebasierten Tests werden auf den Rechnern der Ebene 0 der Universität Bremen durchgeführt. Zum Testen benötigen wir weiterhin eine Entwicklungsumgebung. Hierfür benutzen wir die gleiche Umgebung, die auch zum Erstellen des Systems verwendet wurde. Gruppenintern hat man sich auf die Nutzung der Entwicklungsumgebung NetBeans geeinigt (Version 6.0.1). Für Dokumentationszwecke brauchen wir ein Programm zur Textverarbeitung. Da wir auf verschiedenen Betriebsystemen arbeiten, und die jeweiligen Textverarbeitungsprogramme von Windows,Linux und MacOS nicht zu einander kompatibel sind, haben wir uns für LaTeX entschieden. Gegebenfalls ist die Verwendung von OpenOffice auch legitim, da man auf beiden Betriebsystemen (Windows und Linux) dieses Programm

zur Textverarbeitung installieren kann. Von beiden Programmen wird jeweils die neueste Version benötigt.
Als Testwerkzeug verwenden wir, wie vorgeschrieben, JUnit. JUnit muss nicht extra importiert werden, da in der neuesten Version von NetBeans JUnit integriert ist. JUnit hilft bei der Suche nach Fehlern in den von uns erstellten Klassen und beim Erstellen von Testfällen.
Andere Betriebsmittel, die gebraucht werden, um die Tests durchzuführen, sind z.B. das im Projekt involvierte Personal. Ausserdem wird ein Arbeitsraum benötigt, in dem wir arbeiten können und in dem man Besprechungen machen kann. Hierfür stehen uns die Wohnungen der einzelnen Mitglieder und die Räumlichkeiten der Universität Bremen, inbesondere die Ebene 0 des MZH, zur Verfügung.

8 Testfälle

(bearbeitet von: Artur Malek Yasin Ünsal)

In diesem Abschnitt werden nun die die Testfälle beschrieben. Hier wird genau beschrieben, was von wem und wie getestet wird. Die Tests, die später durchgeführt werden, basieren dann auf den hier beschriebenen Testfällen. Der Fokus dieses Abschnitts liegt auf den Integrations- und Leistungstests. Die Komponententest müssen nicht näher beschrieben werden.

8.1 Komponententests

(bearbeitet von: Artur Malek Yasin Ünsal)

Wie schon oben erwähnt müssen die Komponententests nicht näher beschrieben werden. Auf ihre Spezifikation kann in diesem Abschnitt verzichtet werden. Diese ist durch die JUnit Testfälle gegeben. Die Komponententests wurden schon erstellt und liegen diesem Element bei. Ingesamt wurden bisher 4 Klassen durch Komponententests getestet. Weitere Klassen werden während der Implementierungsphase durch Komponententest getestet. Die White-Box-Tests wurden bisher noch nicht erstellt. Sie werden nach Abschluss der Implementierungsphase erstellt und mit dem fertigen System zusammen abgegeben. Hinzufügend muss man sagen, das der Tester nicht der Autor der Klasse sein darf, da er eine eigene Leseart für seine geschriebene Klasse besitzt und so Fehler leicht übersehen kann.

Klasse	Tester	Implementierer	Testart
Person	Sascha Schmidt	Artur Malek	Black-Box
Project	Artur Malek	Sascha Schmidt	Black-Box
Report	Levent Özenen	Yasin Ünsal	Black-Box
WorkingPackage	Yasin Ünsal	Levent Özenen	Black-Box

8.2 Integrationstests

(bearbeitet von: Yasin Ünsal)

IT-1	Integration Projekt-Arbeitspaket	
IT-1a	Testziel	Löschen eines Arbeitspaketes aus einem Projekt
	Testobjekte	Projekt und Arbeitspaket
	Voraussetzung	Benutzer ist Projektleiter, Projekt enthlt Arbeitspaket
	Umgebung	Testtreiber
	Eingabe	Arbeitspaket löschen
	erwartetes Resultat	Arbeitspaket ist aus dem Projekt gelöscht
	Anforderungen	keine
	Abhängigkeiten	keine

IT-1	Integration Projekt-Arbeitspaket	
IT-1b	Testziel	Hinzufügen eines Arbeitspaketes in ein Projekt
	Testobjekte	Projekt und Arbeitspaket
	Voraussetzung	Benutzer ist Projektleiter
	Umgebung	Testtreiber
	Eingabe	Arbeitspaket hinzufügen
	erwartetes Resultat	Arbeitspaket wurde dem Projekt hinzugefügt
	Anforderungen	keine
	Abhängigkeiten	keine

IT-1	Integration Projekt-Arbeitspaket	
IT-1c	Testziel	Projektleiter editiert ein Arbeitspaket
	Testobjekte	Projekt, Arbeitspaket
	Voraussetzung	Benutzer ist Projektleiter, Projekt hat ein Arbeitspaket
	Umgebung	Testtreiber für die GUI
	Eingabe	Arbeitspaketänderungen
	erwartetes Resultat	editiertes Arbeitspaket
	Anforderungen	keine
	Abhängigkeiten	keine

IT-2	Integration Projekt-Data	
IT-2a	Testziel	Hinzufügen eines neuen Projektes
	Testobjekte	Projekt
	Voraussetzung	Benutzer ist Projektleiter
	Umgebung	Testtreiber
	Eingabe	neues Projekt erstellen
	erwartetes Resultat	Projekt in Projektliste aufgenommen
	Anforderungen	keine
	Abhängigkeiten	keine

IT-2	Integration Projekt-Data	
IT-2b	Testziel	Entfernen eines Projektes
	Testobjekte	Projekt
	Voraussetzung	Benutzer ist Projektleiter
	Umgebung	Testtreiber
	Eingabe	Projekt entfernen
	erwartetes Resultat	Projekt wurde aus der Projektliste entfernt
	Anforderungen	keine
	Abhängigkeiten	keine

IT-2	Integration Projekt-Data	
IT-2c	Testziel	Projektleiter editiert ein Projekt
	Testobjekte	Projekt und Arbeitspaket
	Voraussetzung	Benutzer ist Projektleiter
	Umgebung	Testtreiber
	Eingabe	Projekt anpassen
	erwartetes Resultat	Projekt wurde angepasst
	Anforderungen	keine
	Abhängigkeiten	keine

IT-3	Integration Benutzer-Arbeitspaket	
IT-3a	Testziel	Hinzufügen eines Benutzers zu einem Arbeitspaket
	Testobjekte	GUI, Benutzer und Arbeitspaket
	Voraussetzung	Benutzer ist Projektleiter
	Umgebung	Testtreiber
	Eingabe	Hinzufügen eines Benutzers
	erwartetes Resultat	Benutzer wurde dem Abeitspaket zugewiesen
	Anforderungen	keine
	Abhängigkeiten	keine

IT-3	Integration Benutzer-Arbeitspaket	
IT-3b	Testziel	Löschen eines Benutzers aus einem Arbeitspaketes
	Testobjekte	Benutzer und Arbeitspaket
	Voraussetzung	Benutzer ist Projektleiter
	Umgebung	Testtreiber
	Eingabe	Löschen eines Benutzers aus einem Arbeitspaket
	erwartetes Resultat	Arbeitspaket beinhaltet den User nicht mehr
	Anforderungen	keine
	Abhängigkeiten	keine

IT-4	Integration Benutzer-Data	
IT-4a	Testziel	Löschen eines Benutzers
	Testobjekte	GUI, Benutzer und Data
	Voraussetzung	keine
	Umgebung	Testtreiber
	Eingabe	Löschen eines Benutzers aus dem Benutzerpool
	erwartetes Resultat	Benutzer ist nicht mehr in der Datenbank vorhanden
	Anforderungen	keine
	Abhängigkeiten	keine

IT-4	Integration Benutzer-Data	
IT-4b	Testziel	Hinzufügen eines Benutzers
	Testobjekte	GUI, Benutzer und Data
	Voraussetzung	Benutzer existiert noch nicht
	Umgebung	Testtreiber
	Eingabe	Hinzufügen eines neuen Benutzers in den Benutzerpool
	erwartetes Resultat	Benutzer ist in der Datenbank aufgenommen
	Anforderungen	keine
	Abhängigkeiten	keine

IT-5	Integration Abeitspaket-Benutzer-Zeiterfassung	
	Testziel	Korrekte Zeiterfassung
	Testobjekte	Arbeitspaket, Projekt, Benutzer und Data
	Voraussetzung	Benutzer ist in einem Projekt und hat ein Arbeitspaket angewählt
	Umgebung	Testtreiber
	Eingabe	Starten und Stoppen eines Abeitspaketes
	erwartetes Resultat	Korrekte Zeitangabe im System
	Anforderungen	keine
	Abhängigkeiten	keine

IT-6	Integration Benutzer-Login	
	Testziel	Authentifizieren eines Benutzers
	Testobjekte	GUI, Authentifizierung, Server und Benutzer
	Voraussetzung	Benutzer sendet Login-Anfrage
	Umgebung	Testtreiber
	Eingabe	Anfrage vom Clienten auf Authentifizierung
	erwartetes Resultat	Benutzer ist authentifiziert
	Anforderungen	keine
	Abhängigkeiten	keine

IT-7	Integration Projekt-Import	
	Testziel	Ein Projekt importieren
	Testobjekte	Arbeitspaket, Projekt, GUI und Import
	Voraussetzung	keine
	Umgebung	Testtreiber
	Eingabe	Importieren einer gültigen XML-Datei
	erwartetes Resultat	Projekt wird korrekt in die Projektliste eingefügt
	Anforderungen	keine
	Abhängigkeiten	keine

IT-8	Integration Projekt-Export	
	Testziel	Exportieren eines Projektes
	Testobjekte	Projekt und Export
	Voraussetzung	ein Projekt zum Exportieren existiert
	Umgebung	Testtreiber
	Eingabe	Anfrage von TimeTracker
	erwartetes Resultat	Daten wurden erfolgreich zu TimeTracker exportiert
	Anforderungen	keine
	Abhängigkeiten	Authentifizierung

8.3 Leistungstests

Es gibt vier Leistungstests. Härtetest, Volumentest, Sicherheitstest und Erholungstest. Durch diese Tests werden die die nichtfunktionalen Anforderungen des Systems, wie Robustheit, Performanz, Sicherheit ¨berprüft.

8.3.1 Härtetest

LT-1	die Zugriffe auf die	SQL-Datenbank des Servers
	Testziel	Überprüfung der korrekten Funktionalität bei 10 zeitnahen Anwendungen.
	Voraussetzung	10 Testpersonen fordern die aktuellen Daten der SQL-Datenbank. an.
	Eingabe	GUI des Clients wird upgedated.
	erwartete Ausgabe	hier wird erwartet,dass jeder Client innerhalb von ein paar Sekunden die Daten erhalten (3-5).
	Abhängigkeiten	—-

8.3.2 Volumentest

LT-2	Eingabe der groen Daten	in den Zeiteintrag
	Testziel	Überprüfung der korrekten Funktionalität bei sehr grossen Daten.
	Voraussetzung	Der Benutzer befindet sich auf dem Zeiteintrag (Client).
	Eingabe	Hier werden alle Felder mit der maximalen möglichen Zeicheneingabe in allen Felder ausgefllt und gewartet.
	erwartete Ausgabe	und man erwartet hier, dass alle Angaben vollständig ohne Verlust im Datensatz vorhanden sind.
	Abhängigkeiten	—-

LT-3	mehrere Datensätze	
	Testziel	Überprüfung der korrekten Funktionalität bei 100 Einträgen
	Voraussetzung	es werden 100 zufällig generierte Einträge erstellt
	Eingabe	es werden vom Server zum Client Daten angefordert.
	erwartete Ausgabe	es werden alle Daten übertragen
	Abhängigkeiten	—-

8.3.3 Sicherheitstest

LT-4	Einloggen(Login)	
	Testziel	Überprüfung der korrekten Funktionalität beim Versuch eines Unbefugten Zugriffs
	Voraussetzung	Testpersonen sollten sich im Login befinden
	Eingabe	Eingabe eines falschen Passworts
	erwartete Ausgabe	durch das System werden die Personen wieder aufgefordert, ihren Passwort erneut wiederzugeben.
	Abhängigkeiten	—-

LT-5	die Daten	über das Netzwerk senden
	Testziel	Überprüfung der korrekten Funktionalität beim senden von verschlüsselten Daten
	Voraussetzung	Die Testpersonen wollen wirklich Daten über das Netzwerk senden
	Eingabe	Die Daten, die gesendet werden sollen
	erwartete Ausgabe	die Datenwerden durch das System ent- bzw. verschlüsselt
	Abhängigkeiten	—

LT-6	Offline	
	Testziel	Überprüfung, ob es nur dem Benutzer die Daten bei einer Synchronisation abgeglichen werden
	Voraussetzung	die Testpersonen sollten einen Account und Schreibzugriff auf mindestens ein Element haben.
	Eingabe	es werden nur die gelesenen Daten an der SQL-Datenbank geändert.
	erwartete Ausgabe	es wird ein Fehler auftauchen.
	Abhängigkeiten	—-

8.3.4 Erholungstest

LT-7	Einloggen (Login)	
	Testziel	Überprüfung der korrekten Funktionalität beim Versuch mehrerer unbefugter Zugriffe
	Voraussetzung	Testpersonen sollten sich im Login befinden
	Eingabe	Eingabe eines falschen Passworts
	erwartete Ausgabe	durch das System werden die Personen wieder aufgefordert, ihren Passwort erneut wiederzugeben.
	Abhängigkeiten	—

LT-8	Online und Offline Betrieb	
	Testziel	Überprüfung der korrekten Funktionalität beim mehrmaligen Umschalten
	Voraussetzung	die Testpersonen nutzen den Online–Offline–Betrieb
	Eingabe	die Control-Daten werden durch die Netzwerkverbindung geändert.
	erwartete Ausgabe	es werden hier, korrekte Datensätze auf dem Server und dem Client erwartet.
	Abhängigkeiten	—